Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žan Jaklič

Platforma za prodajo lokalno pridelane hrane

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Problem manjših kmetij, ki se ukvarjajo s pridelavo visoko kvalitetne hrane in pridelka je v doseženi ceni: prodaja preko posrednikov po realni ceni pomeni previsoko končno ceno, ki je kupci ne bi sprejeli. Zato je prihodnost takšnih kmetij v direktni prodaji hrane in pridelka končnim kupcem. Zasnujte platformo, kjer bodo kmetije lahko ponujale in prodajale visoko kvalitetno hrano in pridelek. Kupcem pa omogočale pregled nad ponudbo in nakup po svojih potrebah in željah. Platforma naj bo zasnovana tako, da upošteva oddaljenost ponudnika in kupca.

Zahvaljujem se staršem, ki so me podpirali pri študiju in doc. dr. Roku Rupniku, ki me je usmerjal pri izdelavi diplomske naloge.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	HyperText Markup Language	jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	kaskadne stilske podloge
API	Application Programming Interface	vmesnik za namensko programiranje
SQL	Structured Query Language	strukturirani povpraševalni jezik
CDN	Content delivery network	omrežje za distirbucijo datotek
SMTP	Simple mail transfer protocol	preprost protokol za prenos elektronske pošte
SUPB	Database management system	sistem za upravljanje s podatkovnimi bazami
CLI	Command-line interface	vmesnik z ukazno vrstico
ORM	Object-relational mapping	objektno-relacijsko mapiranje
...	...	...

Povzetek

Naslov: Platforma za prodajo lokalno pridelane hrane

Avtor: Žan Jaklič

Lokalna hrana v zadnjih letih postaja vedno bolj iskana s strani potrošnikov. Zaradi visokih stroškov pridelave kakovostne hrane ta v trgovinah doseže previsoko končno ceno, ki je marsikateri potrošnik ni pripravljen plačati. Zato smo se odločili za razvoj platforme, ki bo predvsem manjšim kmetijam omogočila direktno prodajo hrane končnim kupcem. Z izločitvijo posrednika želimo potrošnikom omogočiti nakup visoko kvalitetne, lokalno pridelane hrane po ugodnih cenah. V prvem delu diplomske naloge smo opisali motiv za izdelavo platforme in obstoječe rešitve. V nadaljevanju je opisan razvoj in delovanje platforme, na koncu pa so navedene še možne izboljšave v prihodnosti in sklepna ugotovitev.

Ključne besede: lokalna hrana, platforma za kmetije, oglaševanje, Django.

Abstract

Title: Platform for selling localy grown crop and food

Author: Žan Jaklič

In the last few years local food became one of the most requested products by consumers. High local food production costs result in high store prices, which consumers are unwilling to pay. For that reason, we decided to develop a platform which offers direct sales of local produce to end customers. With the exclusion of the middleman, we would like to enable consumers to buy high quality local produce at affordable prices. In the first part of the thesis we listed our motive and existing similar platforms. Then we described the development and functionalities of our platform. In the end we gave some thought to possible improvements and made a final conclusion.

Keywords: local food, platform for farmers, advertising, Django.

Poglavje 1

Uvod

V diplomski nalogi smo se posvetili razvoju platforme, ki lokalnim kmetijam omogoča oglaševanje svojih pridelkov, potrošnikom pa naročanje le teh. V naslednjih poglavjih bomo opisali zakaj je taka aplikacija dobrodošla, katere tehnologije smo uporabili za njen razvoj, kako deluje, posamezne postopke razvoja in možne nadgradnje aplikacije, ki smo jo razvili v okviru diplomske naloge.

1.1 Lokalno kmetijstvo

Potrošnik v zadnjih letih izvoru hrane daje vedno več poudarka, med najbolj iskanimi izdelki pa je lokalna hrana. Taka hrana je zaradi krajšega časa transporta manj obdelana, bolj sveža in v večini primerov tudi bolj zdrava. Problem se pojavi, ker lokalna hrana ni ravno poceni, bodisi zaradi velikih marž v trgovinah ali pa zaradi dragih najemnin za stojnice. Nekateri potrošniki zato že kupujejo pridelke direktno na kmetijah, žal pa večina ljudi lokalnih kmetov niti ne pozna.

Tudi v kmetijstvu v Sloveniji velja pravilo ekonomije obsega. To pravilo pravi, da bodo z večanjem količine proizvodnje manjši povprečni stroški pridelave in prodaje, kar velikim kmetijam omogoča boljše možnosti za zaslužek in preživetje. To v praksi okušajo manjši kmetje, ki jih je v Sloveniji vsako

leto manj[1]. Gledano globalno, pa je v primerjavi z nekaterim drugimi državami tudi največji slovenski kmet majhen. Slovenske kmetije morajo zato za uspeh uporabiti drugačno strategijo, kjer je kvaliteta bolj pomembna od kvantitete. Problem pri tem poslovnem modelu pa nastane, ker je proizvodnja izdelkov draga, s prodajo v trgovini pa se prodajna cena še dodatno precej poveča. Velik del ljudi kljub nesporni boljši kvaliteti ni pripravljen plačati toliko denarja za lokalno hrano[2].

1.2 Motiv

Z aplikacijo za lokalna kmetijstva imamo namen povečati prodajo lokalnih slovenskih izdelkov brez trgovine kot posrednika in s tem slovenskim kmetom omogočiti večji dobiček pri prodaji pridelka. Verjamemo, da lahko z osveščanjem potrošnika, kje se nahajajo kmetije in kakšna je njihova ponudba, spodbudimo nakupovanje lokalnih izdelkov cenovno ugodneje direktno na kmetiji.

Namesto dolgega iskanja po spletu lahko potrošnik na naši spletni strani najde ponudbo veliko kmetij na enem mestu. Uporabnik lahko na tedenski ravni prejema tudi elektronska sporočila glede izdelkov, ki ga zanimajo.

S preskokom posrednika v prodajni verigi se bomo ognili dodatnim trgovskim maržam in zmanjšali veliko razliko med ceno lokalnih proizvodov in tistih uvoženih iz eksotičnih držav. Z bolj konkurenčno ceno lokalnega proizvoda bomo marsikaterega kupca prepričali, da vseeno kupi malenkost dražje, vendar bolj zdrave in kakovostne proizvode.

Po drugi strani želimo kmetijam z brezplačnim oglaševanjem omogočiti še en kanal, preko katerega lahko pristopijo do potencialnih kupcev. Z aplikacijo ciljamo predvsem na manjše kmetije, katere se zaradi močne konkurence v zadnjih letih borijo za preživetje.

Poglavje 2

Obstoječe rešitve

Na spletu že obstajajo podobne rešitve, ki skušajo povezati kmetije s potrošniki, vendar pa imajo le te manj funkcionalnosti kot naša spletna aplikacija.

2.1 Lokalna kakovost

- Dostopno na: `http://lokalna-kakovost.si/potrosnike/`

Lokalna kakovost je projekt ministrstva za kmetijstvo, gozdarstvo in prehrano, katerega glavna naloga je promocija lokalne hrane. S projektom želi ministrstvo potrošnikom predstaviti slovenske pridelovalce in jih seznaniti s sistemom sledljivosti živil. S tem nameravajo spodbuditi lokalno samooskrbo in ljudi informirati o prednostih lokalne hrane ter kje jo lahko kupijo[3]. Glavni kanal promocije je družbeno omrežje Facebook, kjer stalno objavljajo članke in intervjuje. Imajo tudi spletno stran na kateri lahko potrošnik in kmet najdeta veliko informacij o lokalni hrani. Posamezen pridelovalec ali predelovalec lahko objavi svoje kontaktne podatke, ki se nato kot lokacija prikažejo na zemljevidu. Kmetije so včasih lahko objavile svoje izdelke in kdaj so le ti na voljo, ne pa tudi dejanskih cen. Zaradi sprememb v aplikaciji so trenutno na voljo samo lokacije pridelovalcev in predelovalcev.

2.2 Kupujmo domače

- Dostopno na: `http://www.kupujmodomace.si/kupdom/index.jsp`

Kmetijsko gozdarska zbornica je leta 2010 izdelala spletno aplikacijo za male oglase, kjer lahko kmetije objavijo svoje izdelke. Njen namen je oglaševalcem omogočiti povečano prodajo izdelkov direktno na kmetiji, potrošnikom pa nakup lokalne hrane. Na strani so prikazani izdelki, ki jih ponujajo določene kmetije, in njihova cena. Ker je stran namenjena samo oglaševanju, izdelkov ne moremo naročiti preko spletne strani. Spletna aplikacija zaradi relativne starosti ne sledi trenutnim grafičnim smernicam, hkrati pa tudi ni prilagojena za mobilne naprave.

2.3 Zeleni krog

- Dostopno na: `http://www.zelenikrog.si/`

Zeleni krog je sestavljen iz skupine kmetij, ki skupaj nudijo široko paleto izdelkov, ki jih vsakih 14 dni potrošnik lahko prevzame na prevzemnih točkah po Sloveniji. Namen društva je kmetom zagotoviti čim večjo prodajo izdelkov v relativno kratkem času na prevzemnem mestu. Kmetje iz naročil opravljenih na spletni strani izvejo točno količino izdelkov, ki jih bodo prodali in s seboj pripeljejo le naročene izdelke. Potrošnik nato izdelke prevzame na prevzemni točki, kjer vsakemu kmetu plača posebej. Ponujeni izdelki so kakovostni in zaradi skupnostnega naročanja tudi cenovno ugodni, vendar pa je interval dostave preredek, da bi potrošnik imel stalen pritok svežih lokalnih izdelkov.

Poglavje 3

Tehnologije

3.1 HTML5

HTML5 je zadnja verzija označevalnega jezika HTML, s katerim definiramo strukturo spletnih strani. HTML strani so sestavljene iz HTML gradnikov, s pomočjo katerih v stran lahko vstavimo konstrukte, kot so slike, obrazci in ostali elementi. Posamezen gradnik je sestavljen iz značk zapisanih v špičastih oklepajih < in >. Večina HTML elementov je sestavljena iz parov značk, začetne in končne . Brskalniki HTML značk ne prikažejo, vendar jih uporabijo za izris vsebine strani. Poleg predstavitve informacij spletnih strani, pa lahko s semantičnimi HTML elementi še dodatno definiramo tudi pomen strani[4].

3.2 CSS3

CSS3 so kaskadne stilske podloge, ki skrbijo za predstavitev spletnih strani. S podlogami lahko določamo atribute kot so barve, pozicije in pisave ter nadziramo aktivnosti, ki jih uporabnik izvaja nad elementi strani. CSS je namenjen predvsem temu, da loči predstavitev in vsebino spletnih strani, kar nam omogoča, da isto HTML stran prikažemo na več načinov. Primer je uporaba podlog za različen prikaz vsebine na računalnikih in mobilnikih [5].

3.3 JavaScript

JavaScript je skriptni programski jezik, ki se ga uporablja za ustvarjanje interaktivnih spletnih strani. Omogoča, da spletne strani reagirajo na dogodke, potrjujejo pravilnost podatkov, zaznajo uporabnikov brskalnik in podobno. JavaScript je bil prvotno namenjen le za implementacijo na odjemalčevi strani, vendar pa je danes vgrajen tudi na strežnikih in v podatkovnih bazah[6].

3.4 Bootstrap

Bootstrap je brezplačno in odprtokodno ogrodje namenjeno oblikovanju spletnih strani. Vsebuje HTML in CSS predloge za pisave, obrazce, gumbe in ostale vmesniške komponente ter JavaScript razširitve. Bootstrap z že zgrajenimi komponentami omogoča hiter razvoj spletnih aplikacij[7].

3.5 Django

Django je ogrodje za razvoj spletnih strani, napisano v jeziku Python. Vsebuje komponente, ki so potrebne na večini spletnih straneh, kar omogoča lažji in hitrejši razvoj spletnih strani, ki uporabljajo podatkovne baze. Do podatkovne baze dostopa prek ORM vmesnika, ki Django razrede pretvori v SQL tabele.
Django sledi konceptu MVT, model, pogled, predloga, ki se od koncepta MVC razlikuje v tem, da sam upravlja kontroler[8]. To je del programske kode, ki skrbi za komunikacijo med modelom in pogledom.
Model predstavlja podatkovno bazo, pogled pa del programske kode, ki sprejme uporabnikov zahtevek in mu nazaj pošlje odgovor. Ostanejo nam še predloge, ki vsebujejo HTML kodo, združeno z jezikom DTL, ki nam omogoča generiranje HTML strani z dinamično vsebino[9].

3.6 Git

Git je sistem za nadzor različic, ki spremlja spremembe v datotekah in na teh omogoča istočasno delo več ljudi. Primarno je namenjen nadzoru in upravljanju sprememb v izvorni kodi pri razvoju programske opreme, vendar pa se ga lahko uporablja za spremljanje sprememb v katerihkoli datotekah. Izvorna koda se z Gitom lahko naloži na GitHub, gostiteljski servis za repozitorije Git. S tem preprečimo morebitno izgubo podatkov, saj je izvorna koda shranjena v oblaku in ne samo na lokalnem računalniku[10].

3.7 Google Maps API

Google Maps API je programski vmesnik, ki razvijalcem omogoča vgrajevanje Googlovih zemljevidov v spletne strani. S pomočjo vmesnika lahko prikazujemo zemljevide, lokacije in pridobivamo geografski položaj uporabnikov. Ponuja prikaz statičnih zemljevidov - slik in dinamičnih zemljevidov, kjer lahko zemljevid premikamo in večamo[11].

3.8 PostgreSQL

Pri razvoju smo se odločili za relacijsko podatkovno bazo. Na voljo smo imeli Djangov privzet SUPB SQLite in ostale brezplačne sisteme za upravljanje s podatkovnimi bazami. Za razvoj podatkovne baze smo se odločili za PostgreSQL, ker omogoča boljšo skalabilnost kot SQLite. Za uporabo PostgreSQL je najprej potrebno spremeniti nekaj nastavitev v datoteki settings.py nato pa se lahko lotimo ustvarjanja podatkovnega modela.
Ko imamo podatkovni model pripravljen lahko z ORM vmesnikom Django razrede pretvorimo v SQL tabele. Po uspešni prvi migraciji moramo še ustvariti skrbnika strani z dostopom do administracijske strani, kjer so prikazani vsi podatki, ki so shranjeni v podatkovni bazi.

3.9 SMTP

SMTP je protokol, ki je namenjen prenosu elektronske pošte[12]. Naša aplikacija nudi obvešanje o novih izdelkih preko elektronskih sporočil, za njihovo pošiljanje pa potrebujemo SMTP strežnik. Za implementacijo naročanja smo uporabili SendGrid, servis za dostavo elektronskih sporočil. SendGrid nudi SMTP strežnike, prek katerih lahko uporabnikom pošiljamo elektronska sporočila.

Poglavje 4

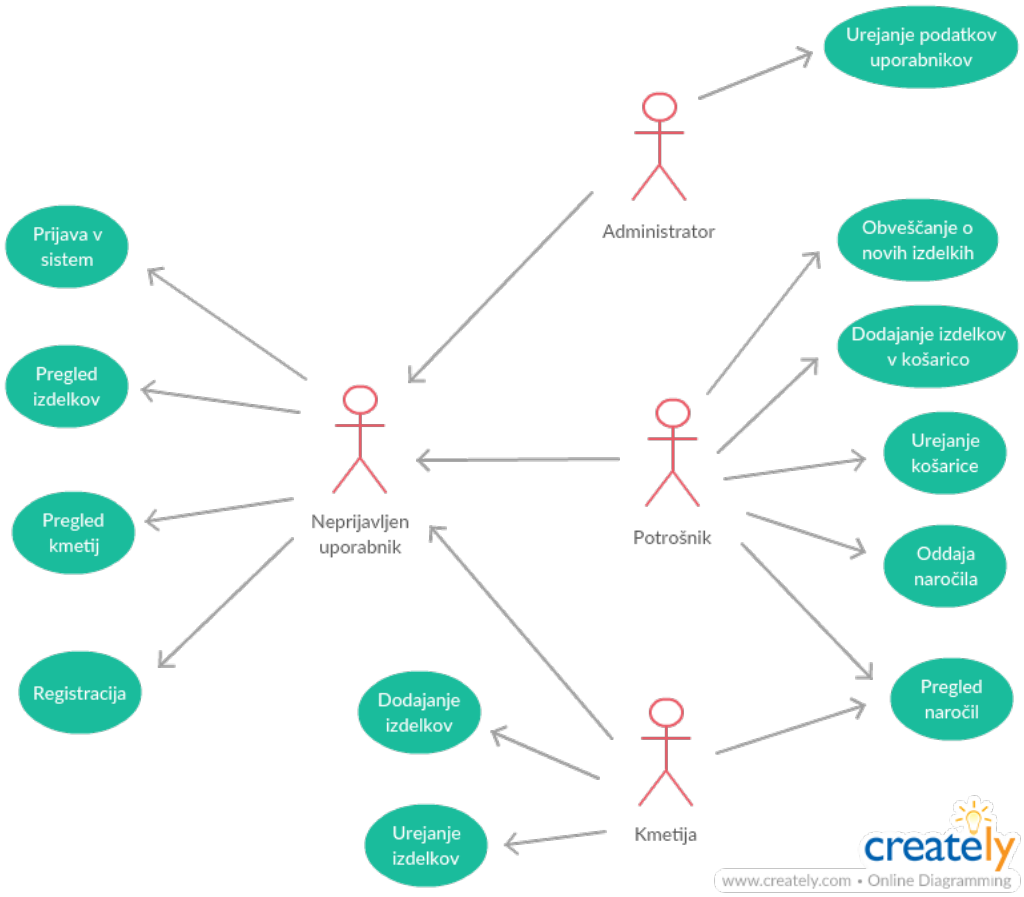
Analiza

4.1 Diagram primerov uporabe

Pred začetkom programiranja spletne aplikacije smo razmislili o vrstah uporabnikov in akcijah, ki jih bodo ti uporabniki izvajali. Cel načrt smo nato realizirali z diagramom primera uporabe (slika 4.1). Kot je na sliki prikazano ima naša aplikacija štiri vrste uporabnikov. Vsi uporabniki imajo dostop do funkcionalnosti, ki so na voljo neprijavljenemu uporabniku ter omejen dostop do funkcionalnosti, ki so omogočene samo določenim uporabniškim vlogam.

4.2 Podatkovni model

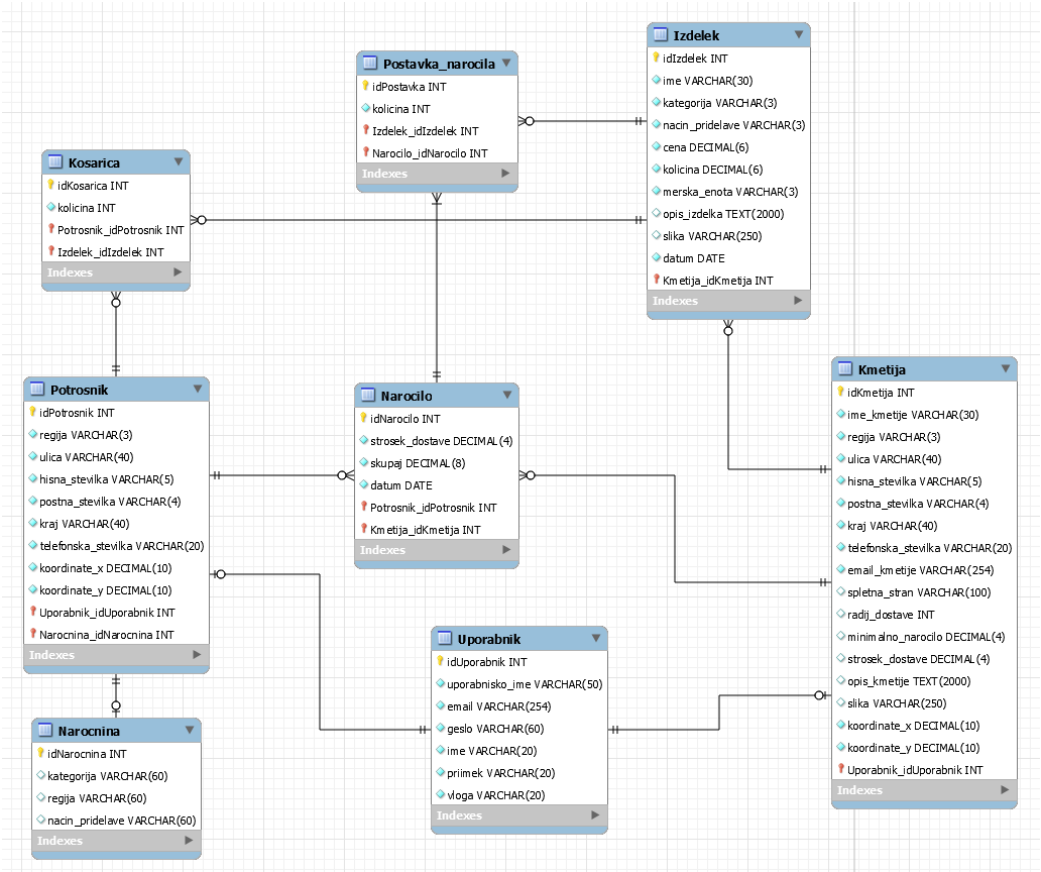
Na začetku razvoja smo si zamislili podatkovni model, ki bo predstavljal tabele v naši podatkovni bazi. Model smo prilagodili glede na že obstoječe tabele ogrodja Django. Odločili smo se, da podatkov zaradi velikega števila poizvedb na bazo ne bomo normalizirali do 3. normalne oblike, ker bi bilo za vsak prikaz izdelka potrebno opraviti več stikov v podatkovni bazi. Podatkovni model smo izdelali že na začetku razvoja, dejansko podatkovno bazo pa smo gradili sproti, ker Django omogoča lahko spreminjanje Python razredov v SQL tabele. Končni podatkovni model, ki smo ga uporabili za izdelavo



Slika 4.1: Diagram primerov uporabe

podatkovne baze je prikazan spodaj (slika 4.2). Podatkovna baza vsebuje osem tabel, ki niso normalizirane do 3. normalne oblike zaradi bolj preproste in hitrejše uporabe v sami aplikaciji.

- **Uporabnik**: osnovni podatki uporabnika. Tabela je že vnaprej pripravljena v ogrodju Django in služi za avtentikacijo uporabnikov, zato je nismo dodatno spreminjali. Poleg gesla je v njej tudi uporabniška vloga, iz katere razberemo do katerih funkcionalnosti aplikacije uporabniku dovolimo dostop.
- **Kmetija**: v tabeli so vsebovani dodatni podatki o uporabnikih, ki



Slika 4.2: Podatkovni model aplikacije

bodo izdelke na naši aplikaciji oglaševali. Poleg kontaktnih podatkov in lokacije lahko dodajo še sliko ter opis kmetije ter podatke o dostavi, če jo omogočajo.

- **Izdelek**: hrani izdelke, ki jih je dodal kmet, na voljo je tudi dodajanje slike
- **Potrošnik**: tabela hrani dodatne podatke, ki specificirajo potrošnika. Med njimi so kontaktni podatki, lokacija in ključi do drugih tabel.
- **Naročnina**: tabela v kateri se nahajajo potrošnikove želje za prejemanje elektronskih sporočil
- **Košarica**: vsebuje vse izdelke, ki jih je uporabnik dodal v košarico.

Namesto v seji je implementirana v podatkovni bazi, da lahko omogočimo enoten prikaz izdelkov na računalniku in mobilni napravi.

- **Naročilo**: tabela naročilo vsebuje zaključena naročila. Vsako naročilo ima natančno enega potrošnika in eno kmetijo. Vrstico v tabeli predstavljajo še strošek dostave, če je ta izbrana, skupen znesek vseh naročenih izdelkov in datum naročila.

- **Postavka naročila**: hrani posamezno postavko naročila, v kateri je poleg ključa za izdelek in naročilo shranjena količina naročenega izdelka.

Poglavje 5

Arhitektura

Slika 5.1: Arhitektura spletne aplikacije

5.1 Odjemalec

Odjemalec je računalniški program, ki dostopa do storitve, tako da pošlje zahtevek nekemu drugemu računalniškemu programu - strežniku. Primer odjemalca je brskalnik, ki s strežnika pridobi spletne strani ali email odjemalec, ki pridobi elektronska sporočila z mail strežnika. V našem primeru je odjemalec internetni brskalnik, ki z uporabnikovega računalnika ali mobilne naprave pošlje zahtevek in počaka, da se mu prikažejo rezultati - spletna stran[13].

5.2 Spletni strežnik

Spletni strežnik je računalniški sistem, ki obdeluje HTTP zahtevke, ki mu jih pošlje odjemalec. Njegova primarna naloga je, da hrani, procesira in dostavlja spletne strani odjemalcem. Za postavitev naše spletne aplikacije smo se odločili za ponudnika oblačnih storitev Heroku, ki uporablja spletni strežnik nginx. Na njem se nahaja aplikacija zgrajena z ogrodjem Django, ki omogoča dinamično grajenje spletnih strani[13].

5.3 Podatkovna baza

Podatkovna baza je organizirana zbirka podatkov. Za razvoj naše spletne aplikacije smo se odločili za relacijsko bazo, ki jo upravlja sistem za upravljanje podatkovnih baz PostgreSQL, ki uporablja jezik SQL. Privzet SUPB ogrodja Django je sicer SQLite, ki je sestavljen iz ene same datoteke vgrajene v končni program. SQLite je preprost za uporabo, vendar pa ima počasne operacije pisanja ter zaradi njegove zasnove ni skalabilen, kar onemogoča večje število uporabnikov aplikacije.

5.4 CDN

CDN je omrežje za distribucijo datotek, ki je sestavljeno iz večih proxy strežnikov in njihovih podatkovnih centrov. Prednost takega omrežja je, da so distribucijski strežniki bližje končnemu uporabniku, rezultat tega pa so krajše zakasnitve pri pošiljanju zahtevkov in prejemanju odgovorov. Za shranjevanje naših statičnih datotek in slik, ki jih naložijo uporabniki, smo se odločili za shranjevanje podatkov v oblaku s storitvijo Amazon Simple Storage System. Skupaj s to storitvijo pa smo uporabili tudi Amazonov CDN CloudFront, ki podatke shranjene v oblaku prenese tudi na druge distribucijske strežnike[14].

Poglavje 6

Razvoj

6.1 Namestitev ogrodja Django

Spletno aplikacijo smo izdelali z, v Pythonu spisanim, ogrodjem Django. Ogrodje smo namestili s pomočjo sistema za upravljanje s paketi, pip, ki skrbi za namestitev programskih paketov napisanih v Pythonu. Namestitev ogrodja Django se izvede z ukazom:

```
1 pip install django
```

Dodatne pakete, ki smo jih potrebovali za razvoj aplikacije smo nameščali vzporedno z razvojem. Po uspešni namestitvi smo z ogrodjem ustvarili projekt, ki predstavlja spletno aplikacijo. Zatem smo ustvarili app, ki v Pythonu opisuje paket, ki zagotavlja funkcije za delovanje aplikacije. Zgornje akcije se izvede z ukazoma:

```
1 django-admin startproject 'Ime projekta'
2 python manage.py startapp 'Ime aplikacije'
```

6.2 Datotečna struktura projekta

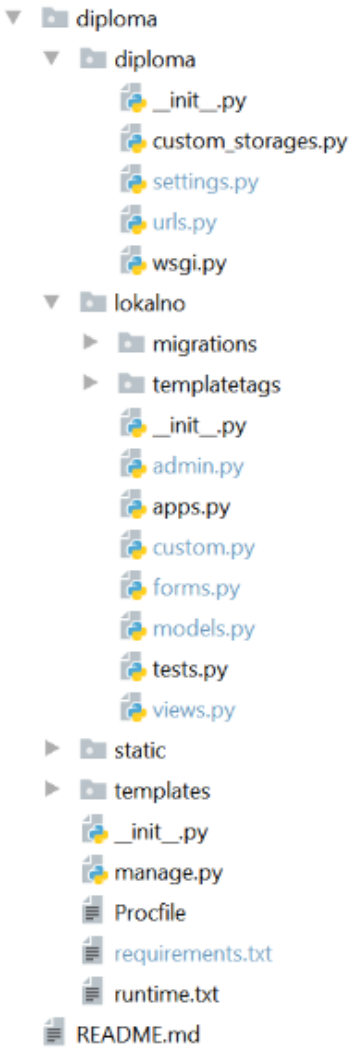
Datotečna struktura Django projekta vsebuje korenski imenik z imenom projekta. V njem so vsebovane aplikacije in datoteke, ki omogočajo delovanje in

upravljanje spletne aplikacije (slika 6.1). Pomembnejše imenike in datoteke bomo spodaj opisali.

- **diploma**: Korenski imenik projekta, ki je hkrati tudi git repozitorij, s katerega izvorno kodo projekta nalagamo na GitHub.
  - **diploma**: Imenik z istim imenom kot projekt, ki vsebuje nastavitve projekta.
    - **settings.py**: V datoteki se nahajajo nastavitve za delovanje aplikacije. To so na primer izvorna datoteka projekta, skrivni ključ za kriptiranje gesla, dolžina seje, poverilnice za povezavo s podatkovno bazo, časovno območje ter naslov in gesla za uporabo storitev Amazon S3 in CloudFront.
    - **urls.py**: Vsebuje shemo z URL vzorci, iz katere strežnik ugotovi, kam mora preusmeriti uporabnika po določenih akcijah.
  - **lokalno**: Direktorij, ki vsebuje Django aplikacijo.
    - **admin.py**: V datoteki so shranjena imena razredov podatkovnega modela, ki se administratorju prikažejo na administracijski strani.
    - **forms.py**: Vsebuje Djangove forme, ki se večinoma pretvorijo v HTML input elemente. Kasneje jih lahko uporabimo za validacijo uporabnikovega vnosa, kjer preverijo ustreznost vnešenih podatkov.
    - **models.py**: Hrani razrede podatkovnega modela, ki jih nato s pomočjo Djanga pretvorimo v SQL tabele.
    - **views.py**: Datoteka, ki hrani funkcije, ki strežniku povejo kako naj reagira na uporabnikov zahteve.
  - **static**: Imenik vsebuje statične datoteke kot so slike, pisave ter CSS in JavaScript kodo.
  - **templates**: V mapi so shranjene HTML datoteke za posamezno stran v aplikaciji. V datotekah se poleg HTML kode nahajajo

tudi spremenljivke, ki jih Django pri procesiranju uporabnikovega zahtevka spremeni v podatke, ki jih pridobi iz podatkovne baze.

– **manage.py**: pomožni program, ki nam v konzoli omogoča izvajanje raznih akcij nad našo aplikacijo.



Slika 6.1: Datotečna struktura projekta

6.3 Avtentikacija uporabnikov

Za shranjevanje uporabnikov smo uporabili Djangov privzeti način, avtentikacijo z uporabniškim imenom in geslom. Pri registraciji se uporabniku glede na izbrano vlogo dodelijo pravice za dostop do določenih delov aplikacije, ki se skupaj z uporabnikovimi podatki shranijo v podatkovno bazo. Za prijavo uporabnikov smo prav tako uporabili že obstoječo funkcijo za avtentikacijo uporabnikov. Vnešeno geslo se po uspešni oddaji obrazva kriptira z zgoščevalno funkcijo in preveri ujemanje z geslom v podatkovni bazi. Po uspešni avtentikaciji se uporabniku odobri dostop do njemu namenjenih delov spletne aplikacije.

6.4 Obrazci

Naša aplikacija vsebuje kar nekaj obrazcev za vnos podatkov, ki se nato prenesejo v podatkovno bazo. Da ne bi prišlo do napačnih ali celo škodljivih vnosov podatkov, se za vsak obrazec preveri ustreznost podatkov. Najprej se podatki preverijo na strani odjemalca prek vnosnih HTML elementov, kjer se lahko določi tip in dovoljeno vsebino vnosa. Vnos se preverja tudi prek JavaScripta, da uporabniku zagotovimo bolj interaktivno izkušnjo, poleg tega pa to pripomore tudi k varnosti. Ko uporabnik odda obrazec, se nato vsi podatki validirajo še na strežniku. Prvo preverjanje na strežniku se opravi preko form, kjer se ponovno preveri pravilnost podatkov, nato pa še v podatkovnem modelu, kjer se podatki ne shranijo, če niso v formatu, ki ga podatkovna baza dovoljuje.

6.5 Spletno gostovanje

Namesto postavitve lastnega spletnega strežnika smo se odločili za oblačno storitev Heroku, ki ponuja brezplačno gostovanje. Razlog za to je odločitev, da bo naša spletna stran na voljo ves čas, česar z lastnim strežnikom ne bi uspeli omogočiti. Za uporabo Heroku storitev moramo najprej namestiti

Heroku CLI, orodje za ustvarjanje in upravljanje Heroku aplikacij. Preden lahko celotno Django aplikacijo naložimo na Heroku moramo ustvariti nekaj datotek, v katerih so podatki, kaj vse naša aplikacija potrebuje za delovanje v oblaku.

- **requirements.txt**: vsebuje imena paketov, ki jih naša aplikacija potrebuje za delovanje
- **Procfile**: deklarira kateri ukaz se mora izvesti za zagon naše aplikacije
- **runtime.txt**: vsebuje Python verzijo, ki bo poganjala našo aplikacijo

Ko imamo pripravljene vse tri datoteke jih skupaj z ostalimi datotekami naložimo na Heroku in tako je naša aplikacija pripravljena za uporabo.

6.6 Shranjevanje datotek v oblaku

Za shranjevanje datotek v oblaku Amazon S3, moramo najprej ustvariti tako imenovano vedro, kamor se shranjujejo vsi naloženi podatku. Zaradi varnostnih razlogov je za upravljanje z vedrom pametno ustvariti novega uporabnika, ki se mu dodelijo le omejene pravice in nima dostopa do drugih veder in storitev. Za omenjenega uporabnika dobimo ključ za dostop do repozitorija in skrivni ključ, ki služi kot geslo, s katerima lahko programsko dostopamo do datotek shranjenih v oblaku.
Naša spletna aplikacija poleg statičnih datotek, ki jih načeloma na strežnik naloži administrator dovoljuje tudi nalaganje slik s strani uporabnikov. Zaradi boljše preglednosti smo se le te odločili shranjevati ločeno, zato smo morali popraviti privzeti način shranjevanja datotek. S sledečo kodo smo paketu, ki pošilja podatke v oblak povedali, naj statične in medijske datoteke pošilja v dva različna direktorija.
Po uspešni nastavitvi skladiščenja smo aktivirali še CDN. CloudFront nam je dodelil lastno domeno preko katere lahko dostopamo do podatkov, shranjenih na Amazon S3. Razlika med storitvama je ta, da se podatki s storitvijo

S3 skladiščijo le na eni izbrani lokaciji, CloudFront pa iste datoteke shrani na več proxy strežnikov. Uporabnik nato po poslanem zahtevku statične in medijske datoteke prejme z najbližjega strežnika.

6.7 Pošiljanje elektronskih sporočil

Za pošiljanje elektronskih sporočil uporabnikom potrebujemo SMTP strežnik. V fazi testiranja sicer lahko ustvarimo lokalni strežnik, ki elektronsko sporočilo izpiše v konzolo, vendar pa rezultati niso primerljivi s pravim elektronskim sporočilom. Odločili smo se za SendGrid, ki nudi enostavno integracijo s storitvijo Heroku. Podobno kot pri shranjevanju datotek v oblak moramo tudi tu pridobiti varnostne poverilnice. S pomočjo uporabniškega imena in gesla lahko prevzamemo univerzalen ključ, ki služi za preverjanje uporabnika.

```
1 sg = sendgrid.SendGridAPIClient('Univerzalen skrivni kljuc')
2 from_email = Email('Posiljatelj')
3 to_email = Email('Prejemnik')
4 subject = 'Zadeva'
5 html_content = 'Vsebina elektronskega sporocila'
6 content = Content('text/html', html_content)
7 mail = Mail(from_email, subject, to_email, content)
8 response = sg.client.mail.send.post(request_body=mail.get())
```

Zgornja koda predstavlja primer pošiljanja elektronskega sporočila s storitvijo SendGrid. V našem primeru smo preprosto besedilo zamenjali s HTML vsebino, ki smo jo prilagodili glede na uporabnikove zahteve. Elektronska sporočila se lahko nato z ukazom v ukazni vrstici strežnika pošilja na tedenski ravni.

Poglavje 7

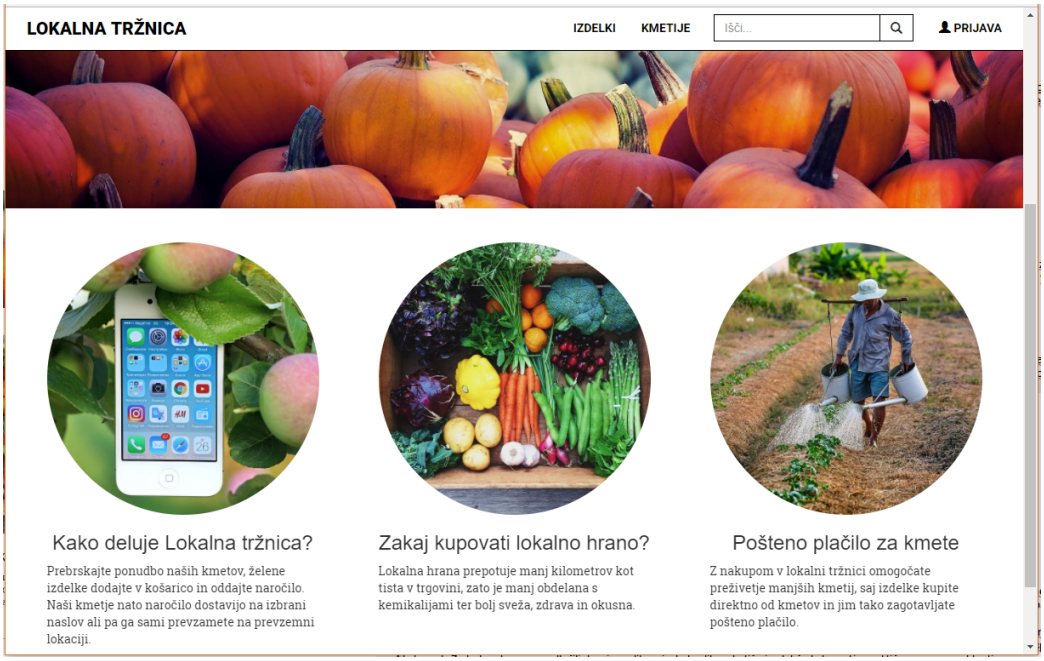
Delovanje aplikacije

7.1 Naslovna stran

Prva stran na kateri uporabnik pristane, ko vpiše url naslov strani, je naslovna stran naše spletne aplikacije. Na naslovni strani smo na kratko opisali kako aplikacija deluje in našteli nekaj prednosti nakupovanja lokalne hrane. Na vsaki strani aplikacije pa je tudi glavni meni, kjer so na voljo povezave do izdelkov, kmetij in prijave (slika 7.1).

7.2 Prijava

Če je uporabnik registriran se lahko v aplikacijo prijavi preko prijavnega obrazca (slika 7.2), kjer mora podati izbrano uporabniško ime ter geslo. Strežnik preveri, če uporabniško ime obstaja, nato pa iz gesla s kriptografsko zgoščevalno funkcijo generira niz, ki ga primerja z nizom, shranjenim v podatkovni bazi. V primeru ujemanja nizov se uporabnika nato prijavi v spletno aplikacijo.
Neregistrirani uporabniki lahko na isti strani izberejo svojo vlogo in se registrirajo prek obrazcev za registracijo.

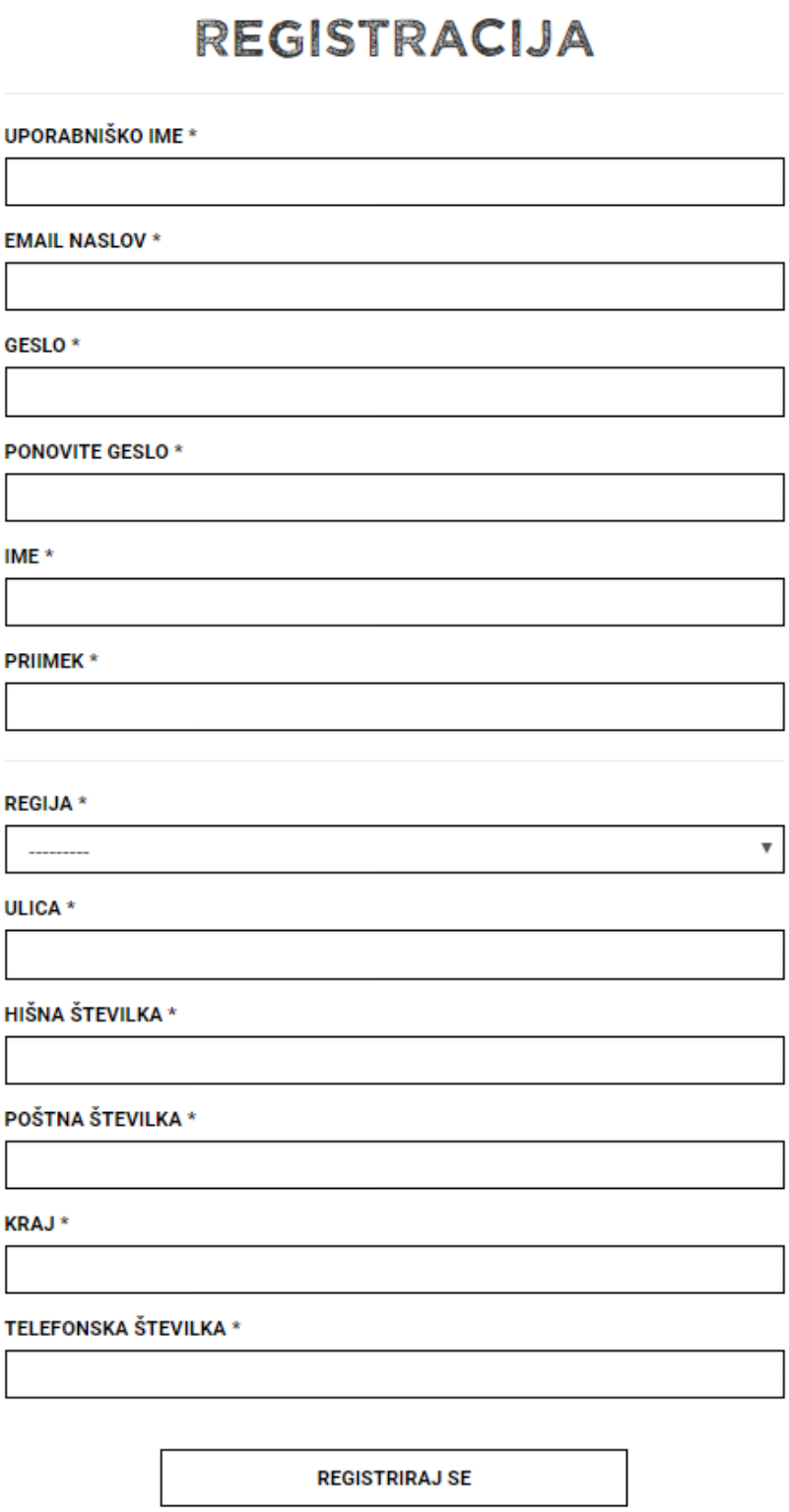


Slika 7.1: Naslovna stran aplikacije

7.3 Registracija potrošnika

Funkcionalnosti prijavljenih uporabnikov se razlikujejo, zato gostu ne moremo omogočiti dostopa do vseh delov spletne aplikacije. Zaradi tega je gostu na voljo registracija, ki se razlikuje glede na izbrano vlogo. Od potrošnika se zahteva vnos osebnih podatkov, ki so potrebni za komunikacijo glede prevzema oziroma dostave naročenih izdelkov (slika 7.3). Po uspešni registraciji strežnik prek Google APIja pridobi geografski položaj uporabnika in ga shrani v podatkovno bazo. Pridobljene koordinate so v aplikaciji kasneje uporabljene za prilagojeno prikazovanje vsebin uporabniku pri iskanju izdelkov in zaključevanju naročila. Hkrati z zaključeno registracijo pa se potrošniku dodajo uporabniške pravice za dostop njemu namenjenih strani.

Slika 7.2: Prijavni obrazec za registrirane uporabnike



Slika 7.3: Registracija za potrošnika

7.4 Urejanje uporabniškega profila

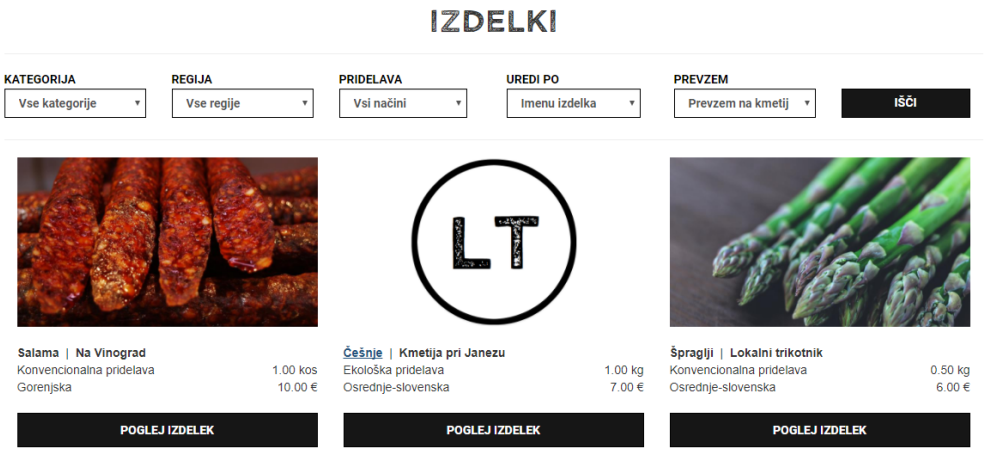
Vsem registriranim uporabnikom je na voljo spreminjanje uporabniškega profila in gesla. Pri spremembi profila je postopek enak kot pri registraciji, kjer se po uspešno oddani formi posodobijo vsi spremenjeni podatki.
Za spremembo gesla more uporabnik najprej vpisati staro geslo, nato pa še dvakrat ponoviti novo.

7.5 Pregled in iskanje izdelkov

Prikaz izdelkov je omogočen za vse vrste uporabnikov, vendar pa je njihov prikaz različen glede na vlogo uporabnika. Prvotno so na strani prikazani vsi obstoječi izdelki, nato pa lahko s pomočjo filtrov pregledujemo specifične artikle, ki nas zanimajo (slika 7.4). Na voljo so možnosti izbire kategorije, regije in načina pridelave izdelkov. Rezultate lahko uredimo glede na ime izdelkov, kmetij ali pa po ceni. Zadnja možnost, način prevzema izdelkov, je na voljo le registriranim uporabnikom, saj je za njeno delovanje potrebna lokacija uporabnika. Uporabniku se glede na njegov geografski položaj prikažejo vsi izdelki, ki so na voljo za dostavo na njegov domači naslov. Iskanje izdelkov pa je na voljo tudi z vnosom niza, kjer se uporabniku prikažejo izdelki, katerih ime, kmetija ali opis se ujema z iskanim nizom.

7.6 Košarica

Poleg pregleda izdelkov, lahko potrošnik želene izdelke dodaja v košarico in jih kasneje naroči. Kmetu je naročanje izdelkov onemogočeno, neprijavljenega uporabnika pa napotimo k obrazcu za prijavo oziroma registracijo. Potrošnik lahko v košarici ureja količino izdelkov, jih odstrani iz košarice, odda naročilo ali pa vse izdelke iz košarice izbriše. Izdelki se v košarici razporedijo glede na to kateri kmetiji pripadajo (slika 7.5). Privzeti način prevzema izdelkov je prevzem na kmetiji, ki izdelke ponuja. Če kmetija nudi dostavo na naslov potrošnika, ima ta tudi možnost izbire dostave, vendar le, če je znesek

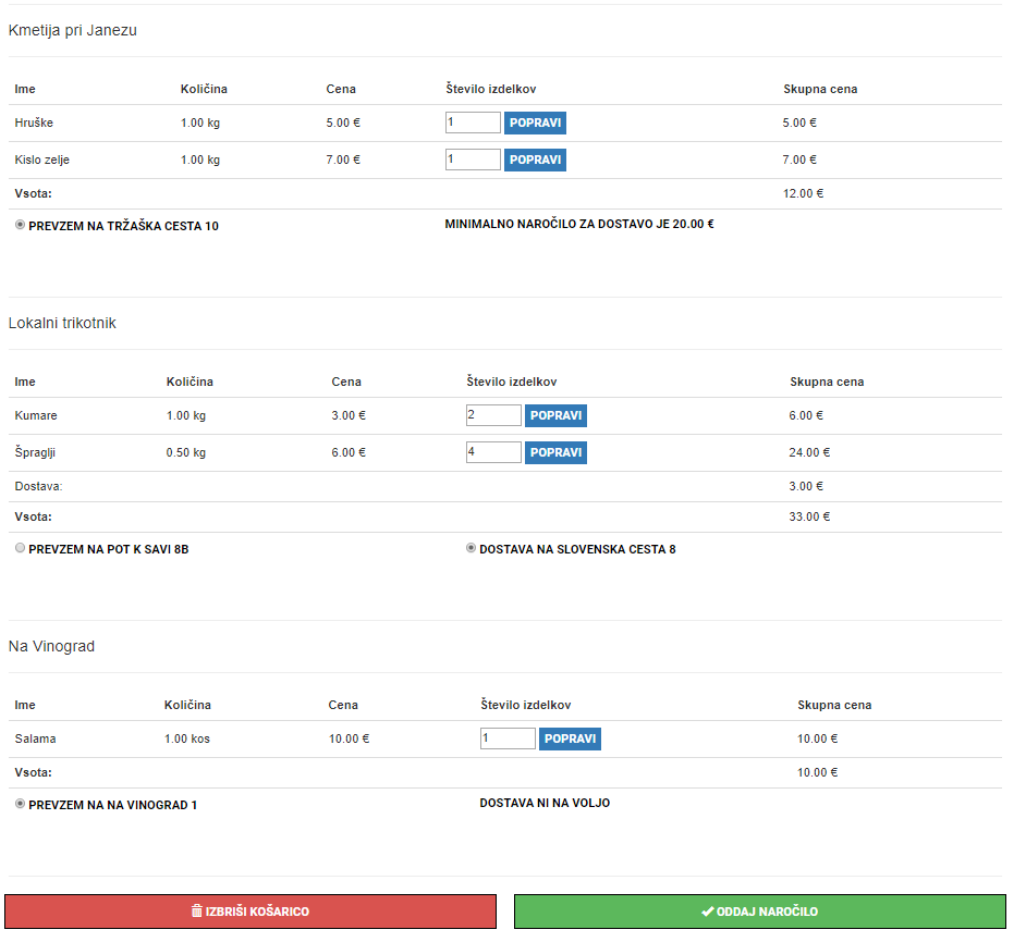


Slika 7.4: Prikaz izdelkov

naročila večji od minimalne vrednosti naročila za dostavo. Če se potrošnik odloči za dostavo na dom, potem se njegovemu naročilu doda še strošek dostave. Ko je uporabnik zadovoljen z izdelki v košarici, zaključi naročilo. Ker mora potrošnik vsakemu kmetu plačati posebej, kmet pa na naročilu ne potrebuje izdelkov drugih kmetij, se za vsako kmetijo ustvari ločeno naročilo.

7.7 Pregled naročil

Potrošnik lahko po oddaji naročila, podrobnosti naročila pregleda. Posamezno naročilo vsebuje kmetijo in njene kontaktne podatke, datum in ceno naročila ter vse izdelke, ki temu naročilu pripadajo (slika 7.6). Pri kmetu je prikaz naročil podoben kot pri potrošniku. Kmet vidi le naročila uporabnikov, ki vsebujejo izdelke z njegove kmetije. Podatki potrošnikov niso javni dokler le ta ne opravi nakupa. Pri vsakem naročilu so kmetu predstavljeni le osebni podatki potrošnika, ki so potrebni za komunikacijo, kot so ime in priimek, naslov ter telefonska številka. Na ta način kmetu priskrbimo podatke za kontakt in dostavo naročenih izdelkov.



Slika 7.5: Košarica

Slika 7.6: Pregled naročil

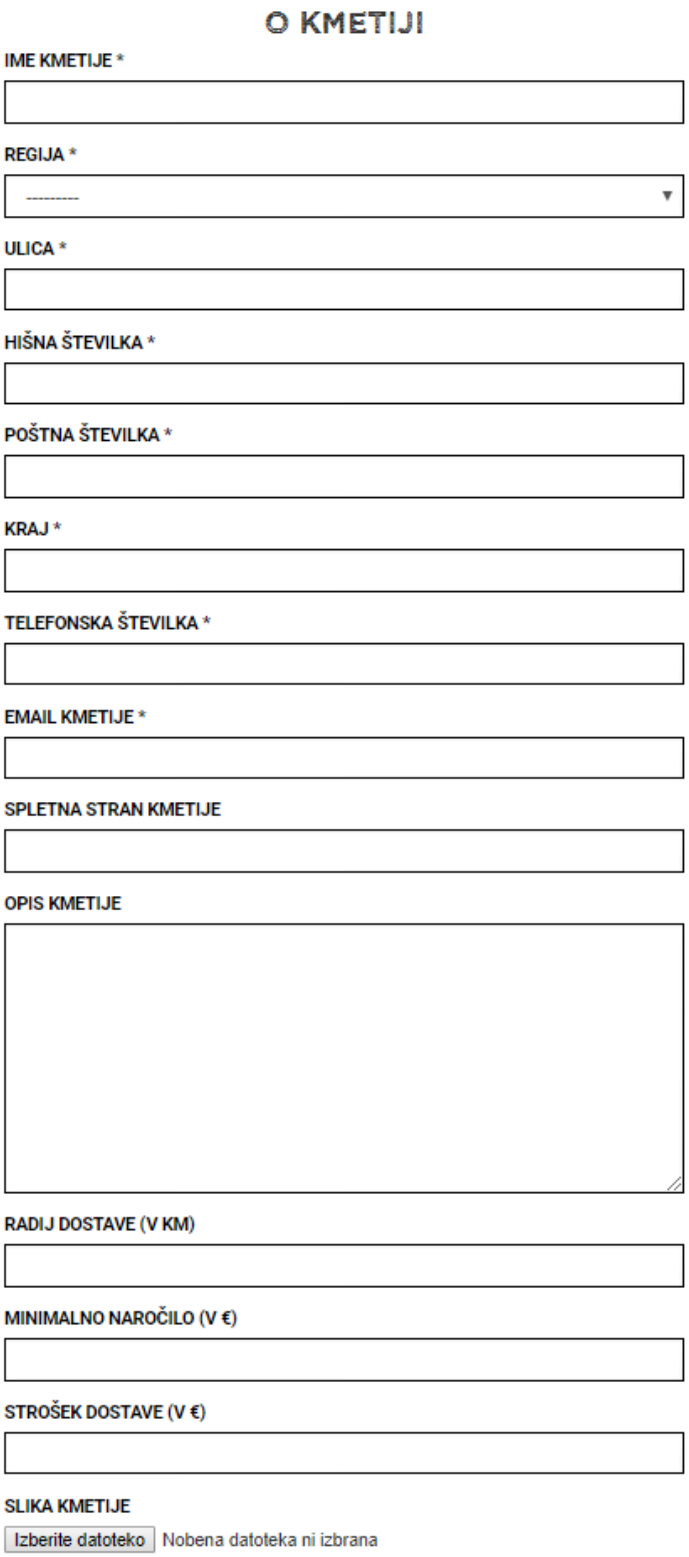
Slika 7.7: Kriteriji za filtriranje izdelkov pri obveščanju

7.8 Obveščanje o novih izdelkih

Potrošniku je omogočena možnost obveščanja o novih izdelkih. Izbere lahko informiranje na podlagi kategorij izdelkov, regij iz katerih kmetije prihajajo in načina pridelave. Potrošniku se nato na podlagi izbire pošlje elektronsko sporočilo o novih izdelkih, ki se ujemajo z njegovo izbiro. Če je potrošnik nezadovoljen ali pa se mu zdi, da dobiva preveč elektronskih sporočil, se lahko od novic odjavi. V primeru, da se želi naročiti še na več izdelkov pa lahko izbiro izdelkov tudi ureja.

7.9 Registracija kmetije

Enako kot pri potrošniku so nekatere funkcije dostopne samo kmetom. Uporabnik, ki pri registraciji izbere vlogo kmetije mora poleg osnovnih osebnih podatkov vnesti tudi podatke o kmetiji (slika 7.8), ki bodo na spletni strani prikazani uporabniku. Kmet ima možnost vnesti tudi opis in naložiti sliko kmetije. V primeru, da kmet nudi dostavo, lahko vnese radij dostave, minimalno vrednost naročila in ceno dostave.



Slika 7.8: Registracija za kmetije

7.10 Prikaz kmetij

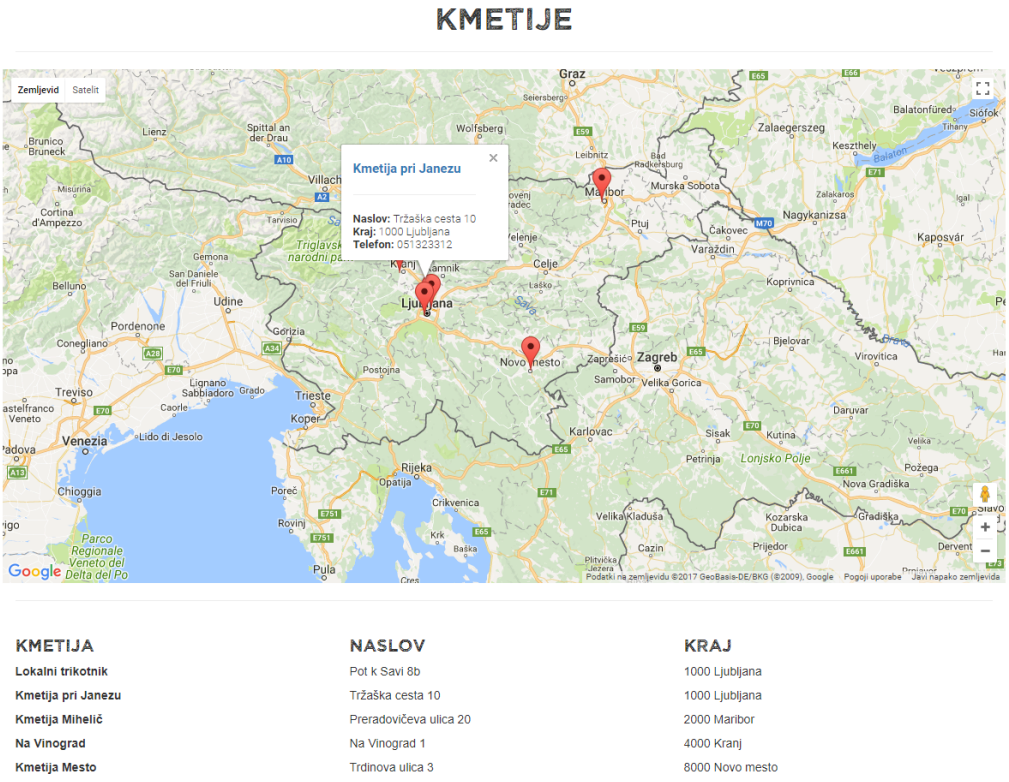
Uporabniku je na voljo zemljevid, na katerem so označene vse lokacije kmetij, ki oglašujejo preko naše spletne aplikacije (slika 7.9). Zemljevid omogoča, da uporabniki hitro vidijo kmetije, ki so v njihovi bližini. S klikom na značko se odpre oblaček s podatki kmetije. Pod zemljevidom so navedene tudi vse kmetije urejene po poštnih številkah. Za dvojni prikaz podatkov smo se odločili, ker je drugi bolj preprost in omogoča hitrejše iskanje kmetij. S klikom na kmetijo se uporabniku prikažejo vsi podatki kmetije, njena slika in točna lokacija kmetije na približanem statičnem zemljevidu. Na isti strani je prikazano tudi število izdelkov kmetije in vsi izdelki, ki jih kmetija ponuja.

7.11 Dodajanje izdelkov

Kmet se od potrošnika razlikuje v tem, da izdelke prodaja in ne kupuje. Zato je kmetu nakupovanje izdelkov onemogočeno, tako kot je potrošniku onemogočeno dodajanje izdelkov v spletno aplikacijo. Kmetu je v aplikaciji na voljo pregled izdelkov, kjer lahko izdelke dodaja, spreminja in briše (slika 7.10). Glavni cilj aplikacije je oglaševanje izdelkov in njihova prodaja potrošnikom, zato pri dodajanju izdelkov spodbujamo, da jih kmetije predstavijo s sliko dejanskih izdelkov in krajšim opisom, kar pripomore k boljši predstavitvi in na koncu tudi prodaji oglaševanih izdelkov.

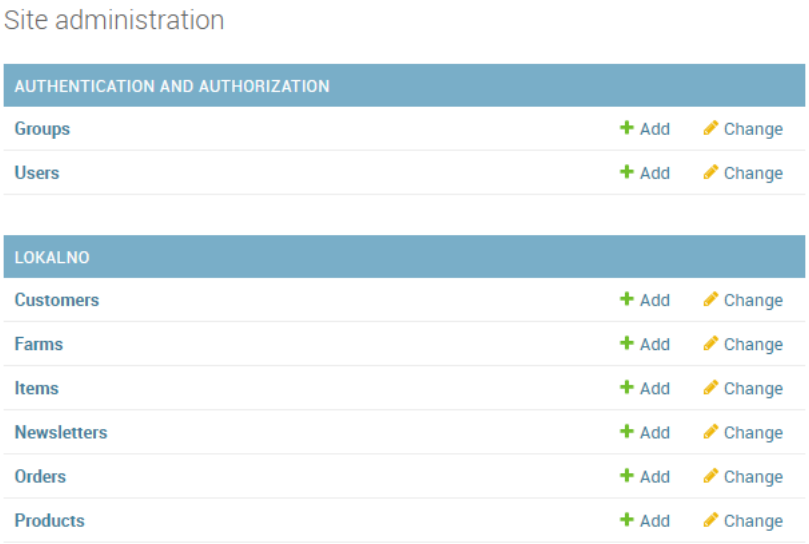
7.12 Spreminjanje podatkov uporabnikov

Administrator ima na voljo dostop do administracijske strani. Ob uspešni avtentikaciji se mu prikažejo podatki iz podatkovne baze (slika 7.11). Skrbnik strani lahko uporabnikom dodaja pravice za dostop do aplikacije. V primeru,



Slika 7.9: Prikaz kmetij

Slika 7.10: Pregled izdelkov kmetije



Slika 7.11: Prikaz tabel iz podatkovne baze

ko uporabnik sam ne more spremeniti kakšnih podatkov ali pa so njegovi podatki neustrezni jih administrator lahko popravi ali izbriše.

Poglavje 8

Optimizacija

8.1 Optimizacija slik

Naša aplikacija na glavni strani prikazuje veliko slik, ki so jih naložili uporabniki. Zato je potrebno, da velikost slik čimbolj pomanjšamo in optimiziramo, da uporabniki aplikacije na mobilnih telefonih ne porabijo preveč mobilnih podatkov. Problem slik naloženih iz uporabnikove strani pa je tudi ta, da so v večini vse različnih velikosti in razmerij. Da bi omogočili enoten prikaz vseh slik na spletni aplikaciji, smo se odločili za najbolj pogosto razmerje slik 16:9 in vse slike pretvorili v te dimenzije. Prevelike slike smo zmanjšali, premajhne povečali, nato pa jih iz sredine navzen obrezali, da so ustrezale želenim dimenzijam. Slikam smo spotoma tudi zmanjšali kakovost. Za ta korak smo se odločili, ker je razlika v izgledu slike z boljšo in slabšo kakovostjo praktično neopazna, hkrati pa precej zmanjša velikost same slike. Z optimizacijo slik smo uporabnikove slike z velikostjo par MB uspeli spraviti na približno 40 KB.

8.2 Minimizacija kode

Vse CSS in JavaScript datoteke smo najprej združili v eno posamezno datoteko za CSS in eno za JavaScript, da smo zmanjšali število HTTP zahtevkov

na strežnik. Zatem smo združeno kodo še minimizirali s pomočjo za to namenjenih orodij. Minimizacija kode je postopek, pri katerem iz kode odstranimo nepotrebne znake kot so presledki, nove vrstice in komentarji. Z minimizacijo kode zmanjšamo velikost same datoteke in s tem pohitrimo njen prenos[15].

Poglavje 9

Testiranje

Naša spletna aplikacija je namenjena tako za uporabo na računalnikih kot na mobilnih telefonih. Razvijali smo jo z namenom, da bo prikaz vsebine primeren za vse naprave. Izgled in delovanje naše aplikacije smo zato na računalniku testirali v brskalnikih Google Chrome, Mozilla Firefox in Microsoft Edge. Za zagotovitev čim boljše izkušnje na vseh mobilnih telefonih smo najprej na računalniku testirali več različnih resolucij, ki jih uporabljajo mobilni telefoni, na koncu pa aplikacijo testirali še na mobilnem telefonu in tablici.

Poglavje 10

Nadaljni razvoj

10.1 Povratne informacije uporabnikov

Prva stvar, ki jo imamo namen pridobiti v prihodnosti, so povratne informacije uporabnikov, katerim je aplikacija namenjena. V praksi bi morali zahteve pridobiti že v postopku analize, vendar pa bi v našem primeru za dobro predstavo morali kontaktirati kar precej ljudi, zato smo se odločili, da bomo to storili naknadno. Smiselno se nam zdi, da lahko kmetom in uporabnikom pošljemo povezavo do že izdelane aplikacije, kjer lahko njene funkcionalnosti bolj podrobno pregledajo in komentirajo. Z njihovo pomočjo bomo poskušali aplikacijo izboljšati in jo čim bolje približati končnemu uporabniku.

10.2 Celostna grafična podoba

Skozi razvoj aplikacije smo se trudili upoštevati načela dobrega oblikovanja, vendar pa zaradi pomanjkanja našega znanja na tem področju ostaja še veliko prostora za izboljšave. Možne izboljšave celostne grafične podobe so izdelava novega logotipa in izbira barvne palete, ki dobro predstavi stran in je povezana z vsebinami aplikacije.

10.3 Uvoz izdelkov v podatkovno bazo

Nekatere kmetije že oglašujejo svoje izdelke na lastnih spletnih straneh ali pa na družbenih omrežjih kot je Facebook. Da bi jim zagotovili čim boljšo uporabniško izkušnjo s čim manj potrebnega dela, imamo namen razviti tudi vmesnik za uvoz izdelkov v podatkovno bazo aplikacije. Podpreti mislimo uvoz izdelkov v formatu XML in JSON, ki sta najpogosteje uporabljena.

10.4 Mobilna aplikacija

Zaradi večanja števila mobilnih naprav je vedno več uporabnikov na splet povezanih prek mobilnih telefonov. Čeprav smo našo aplikacijo že prilagodili za uporabo na mobilnih napravah, pa je razlika med delovanjem spletne in mobilne aplikacije še vedno precejšnja. V bližnji prihodnosti želimo zato izdelati tudi mobilno aplikacijo s podobnimi funkcionalnostmi.

10.5 Zanesljivost kmetov

V nadaljnem razvoju imamo namen implementirati tudi sistem za ocenjevanje in komentiranje kmetij ter izdelkov. S tem želimo zaščititi potrošnika, hkrati pa nagraditi kmetije, ki poslujejo dobro in kaznovati nezanesljive kmete, ki ne dostavljajo izdelkov.

10.6 Plačilne metode

Večina spletnih strani, ki se ukvarja s prodajo izdelkov podpira različne plačilne metode kot sta plačevanje s kreditnimi karticami in PayPal. Problem naše aplikacije je predvsem ta, da ne bomo imeli le enega računa kamor se stekajo plačila, ampak bo vsaka kmetija imeli svojega. Iz tega razloga bomo morali pazljivo implementirati sistem, kjer bomo pri vnosu računov kmetij morali podrobno preveriti verodostojnost podatkov.

10.7 Optimizacija za iskalnike

Naša aplikacija za uspešno delovanje potrebuje veliko potrošnikov kot tudi kmetij, zato je pomembno, da jo na spletu lahko najde čim več ljudi. Skozi razvoj smo izvorno kodo že prilagajali za iskalnike. Da bi čim boljše prilagodili tudi besedila spletne aplikacije, bomo v anketah spraševali ljudi na kakšne načine iščejo lokalno hrano in kakšne izraze vnašajo v iskalnike. Po analizi rezultatov bomo nato besedila naše spletne aplikacije prilagodili za čim boljše rezultate v iskalnikih.

Poglavje 11

Sklep

Cilj diplomske naloge je bil razvoj spletne aplikacije, ki bo povezala kmete in potrošnike ter tako omogočila nižjo ceno lokalnih pridelkov. V Sloveniji sicer že obstaja nekaj tovrstnih aplikacij, ki pa niso moderne, funkcionalno celovite in prilagojene mobilnim napravam. Zato je razvoj spletne aplikacije, kot dodatnega prodajnega kanala, ki ga predlagamo v diplomski nalogi vsekakor smiseln. Problem naše aplikacije je ta, da mora za uspešnost imeti veliko število uporabnikov, tako kmetov kot tudi potrošnikov. Da bi bilo to mogoče doseči menimo, da jo je potrebno opremiti z omenjenimi nadgradnjami, da bo bolj zanimiva za uporabnike. Dolgoročne nizke cene izdelkov je na žalost možno obdržati le na dva načina, ker razvoj in vzdrževanje aplikacije nista trivialna in brezplačna, z zaračunavanjem stroškov aplikacije uporabnikom pa bi mi postali novi posrednik, česar si ne želimo. Prva možnost je, da nadgradnjo aplikacije financira država kar trenutno na nekaterih projektih že počne, druga pa je financiranje razvoja iz oglasov, kar pa je mogoče le pri velikem številu uporabnikov.

Trenutno stanje spletne aplikacije omogoča uporabo, kot je bila zamišljena na začetku projekta, vendar pa se zavedamo, da je aplikacijo možno še dodatno izboljšati, kar nameravamo storiti v prihodnosti.

Literatura

[1] "Število kmetijskih gospodarstev." Dosegljivo: http://www.stat.si/StatWeb/News/Index/6208. [Dostopano: 2. 9. 2017].

[2] "Zakaj izbrati lokalno hrano?." Dosegljivo: http://lokalna-kakovost.si/izbrati-lokalno-hrano/. [Dostopano: 2. 9. 2017].

[3] "Lokalna kakovost - naša super hrana." Dosegljivo: https://www.facebook.com/pg/nasasuperhrana/about/. [Dostopano: 4. 9. 2017].

[4] "Html." Dosegljivo: https://en.wikipedia.org/wiki/HTML. [Dostopano: 1. 9. 2017].

[5] "Cascading style sheets." Dosegljivo: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. [Dostopano: 1. 9. 2017].

[6] "Javascript." Dosegljivo: https://en.wikipedia.org/wiki/JavaScript. [Dostopano: 1. 9. 2017].

[7] "Bootstrap." Dosegljivo: https://getbootstrap.com/. [Dostopano: 1. 9. 2017].

[8] "Django mvc." Dosegljivo: https://www.tutorialspoint.com/django/django_overview.htm [Dostopano: 1. 9. 2017].

[9] "Django." Dosegljivo: https://www.djangoproject.com/. [Dostopano: 1. 9. 2017].

[10] "Git." Dosegljivo: https://en.wikipedia.org/wiki/Git. [Dostopano: 1. 9. 2017].

[11] "Google maps api." Dosegljivo: https://developers.google.com/maps/faq. [Dostopano: 1. 9. 2017].

[12] "Simple mail transfer protocol." Dosegljivo: https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol. [Dostopano: 4. 9. 2017].

[13] "Client-server model." Dosegljivo: https://en.wikipedia.org/wiki/Client-server_model. [Dostopano: 1. 9. 2017].

[14] "Content delivery network." Dosegljivo: https://en.wikipedia.org/wiki/Content_delivery_network. [Dostopano: 1. 9. 2017].

[15] "Minification." Dosegljivo: https://en.wikipedia.org/wiki/Minification_(programming). [Dostopano: 1. 9. 2017].

[16] J. Duckett, *HTML and CSS : Design and Build Websites*. Wiley, 2011.

[17] J. Duckett, *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley, 2014.

[18] J. Elman and M. Lavin, *Lightweight Django: Using REST, WebSockets, and Backbone*. O'Reilly, 2015.